# Curso de Introdução a Python
# Aula 04: Estruturas de decisão

Ana Luiza Martins Karl

# Sumário

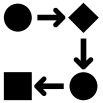
# 01

# Introdução

- Estruturas de decisão
- Condições

# Introdução

**Programa: conjunto de instruções a serem executados pela máquina**

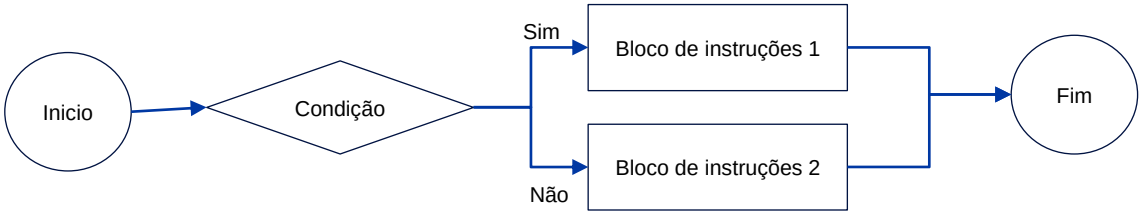


**Python: linguagem de programação interpretada: execução linha a linha**

**Estruturas de decisão: permite a escolha das instruções a serem executadas**

# Introdução

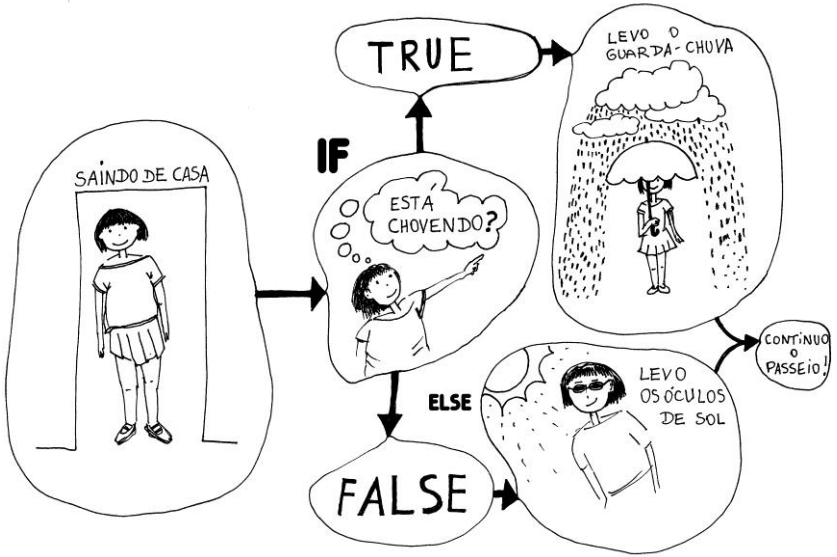
**Nas estruturas de decisão, o fluxo de instruções a ser seguido é escolhido em função do resultado da avaliação de uma ou mais condições.**




**Uma condição é uma expressão lógica.**

# Condições

**Uma condição é uma expressão lógica. Logo, o resultado de uma condição é do tipo booleano**: **True** ou **False**



# 02

# Operadores

- ==; !=; <; >; <=; >=
- is; is not
- in; not in

# Operadores de atribuição

O Python herda dos mesmos operadores aritméticos da matemática:

- **+** - adição;
- **-** - subtração;
- ***** - multiplicação;
- **/** - divisão;
- **//** - divisão inteira;
- **%** - resto;
- ****** - exponenciação
- **+=** - x = x + 1
- **-=** - x = x - 1

# Operadores de comparação

O Python herda dos mesmos operadores aritméticos da matemática:

- **==** - igual a ;
- **!=** - diferente de;
- **<** - menor que;
- **>** - maior que;
- **<=** - menor ou igual que;
- **>=** - maior ou igual que;

**Atenção**: É possível comparar dados de tipos diferentes. Entretanto, nem sempre os resultados são intuitivos.

# Operadores de identidade

Para comparar objetos, verificamos se ambos os objetos testados referenciam o mesmo objeto:

- **is** – retorna ***True*** se os objetos são os mesmos
- **is not** – retorna ***True*** se os objetos não são os mesmos

```
lista = [1, 2, 3]
outra_lista = [1, 2, 3]
recebe_lista = lista

# Recebe True, pois são o mesmo objeto
print(f"São o mesmo objeto? {lista is recebe_lista}")

# Retorna False, pois são objetos diferentes
print(f"São o mesmo objeto? {lista is outra_lista}")
```

# Operadores de associação

Os operadores de associação servem para verificar se determinado objeto ou valor está associado ou pertence a determinada estrutura de dados

- **in** – retorna ***True***, caso o valor seja encontrado na sequência
- **not in** – retorna ***True***, caso o valor seja encontrado na sequência

```
lista = ["Python", 'Academy', "Operadores", 'Condições']

# Verifica se existe a string dentro da lista
print('Python' in lista)  # Saída: True

# Verifica se não existe a string dentro da lista
print('SQL' not in lista) # Saída: True
```

# Operadores lógicos

Os operadores lógicos possibilitam construir os testes lógicos:

- **and** – retorna **True** se ambas as afirmações forem verdadeiras
- **or** – retorna **True** se uma das afirmações for verdadeira
- **not -** retorna **Falso** se o resultado for verdadeiro

```
num1 = 7
num2 = 4

# Exemplo and
if num1 > 3 and num2 < 8:
    print("As Duas condições são verdadeiras")

# Exemplo or
if num1 > 4 or num2 <= 8:
    print("Uma ou duas das condições são verdadeiras")

# Exemplo not
if not (num1 < 30 and num2 < 8):
    print('Inverte o resultado da condição entre os parânteses')
```

# 03

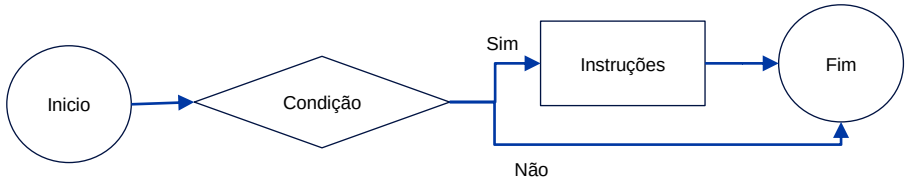
# Estruturas de condição do tipo “se”

- Condicionais simples
- Condicionais compostas
- Condicionais encadeadas

# Se – a estrutura *if* simples

A condicional ***if*** pode ser escrita da seguinte maneira:

```
if (condição):
    Instruções
```

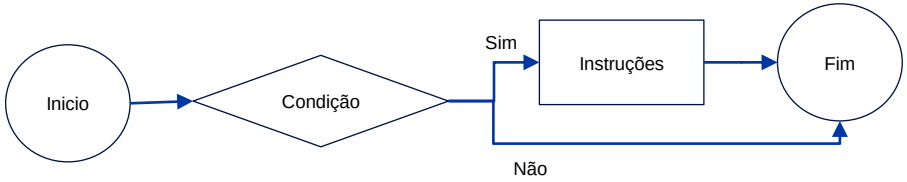



**Atenção**: Python depende de indentação (espaços em branco no início de uma linha) para definir o escopo no código.

# Se – a estrutura *if* simples

A condicional ***if*** pode ser escrita da seguinte maneira:

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

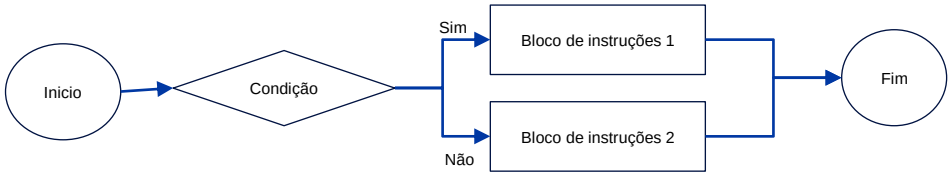



**Atenção**: Python depende de indentação (espaços em branco no início de uma linha) para definir o escopo no código.

# Se – a estrutura *if* composta

O ***else*** é utilizado para criar um segundo bloco de instruções, que será executado caso a condição ***if*** não seja satisfeita.
Podemos escrever da seguinte maneira:

```
if (condição):
    Instruções
else:
    Instruções
```

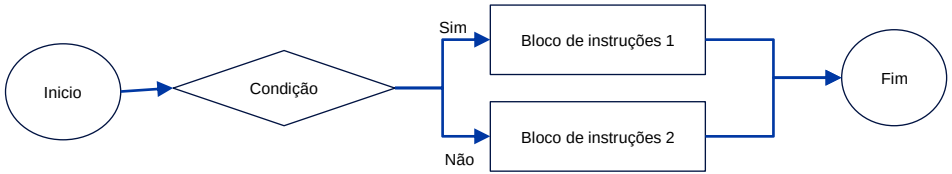



**Atenção**: Python depende de indentação (espaços em branco no início de uma linha) para definir o escopo no código.

# Se – a estrutura *if* composta

O ***else*** é utilizado para criar um segundo bloco de instruções, que será executado caso a condição ***if*** não seja satisfeita.
Podemos escrever da seguinte maneira:

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

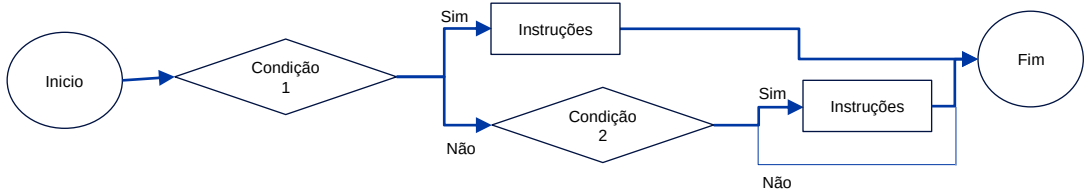



**Atenção**: Python depende de indentação (espaços em branco no início de uma linha) para definir o escopo no código.

# Se – a estrutura *if* encadeada

A palavra-chave "***elif***" é a maneira do Python de dizer "se as condições anteriores não foram verdadeiras, então tente esta outra condição".

```
if (condição 1):
    Instruções
elif (condição 2):
    Instruções
else:
    Instruções
```

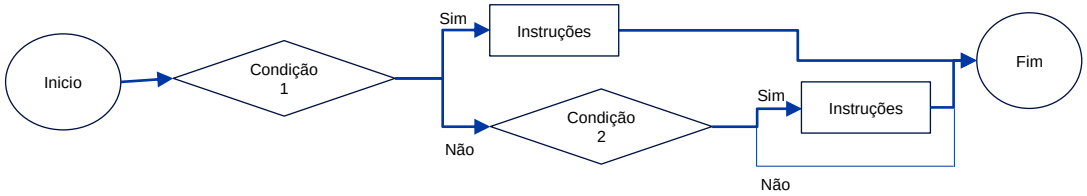



**Atenção**: Python depende de indentação (espaços em branco no início de uma linha) para definir o escopo no código.

# Se – a estrutura *if* encadeada

A palavra-chave "***elif***" é a maneira do Python de dizer "se as condições anteriores não foram verdadeiras, então tente esta outra condição".

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```



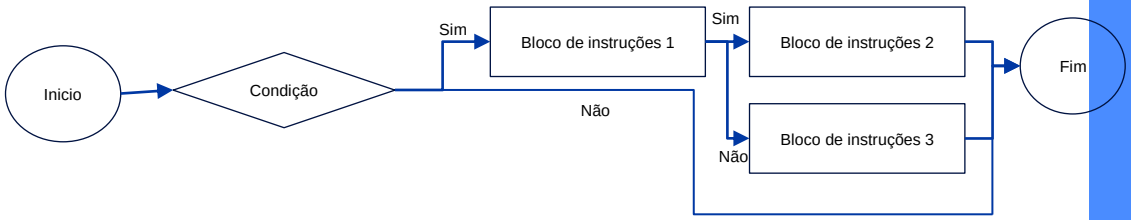

**Atenção**: Python depende de indentação (espaços em branco no início de uma linha) para definir o escopo no código.

# Se – a estrutura *if* aninhada

Você pode ter declarações ***if*** dentro de declarações ***if***, isso é chamado de declarações ***if*** aninhadas.

```
x = 41

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```




**Atenção**: Python depende de indentação (espaços em branco no início de uma linha) para definir o escopo no código.

# Se – a estrutura *if* curta

Se você tem apenas uma instrução para executar, você pode colocá-la na mesma linha da instrução ***if***.

```
if a > b: print("a is greater than b")
```

Usando o ***else***:

```
a = 2
b = 330
print("A") if a > b else print("B")
```

Semelhante ao **operador ternário** em C.

# Se – combinando condições

Operadores lógicos: **not**, **and** e **or**

```
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")
```

# Se – combinando condições

Operadores lógicos: **not**, **and** e **or**

```
a = 33
b = 200
if not a > b:
  print("a is NOT greater than b")
```

# Se – combinando condições

Operadores lógicos: **not**, **and** e **or**

```
a = 200
b = 33
c = 500
if a > b or a > c:
  print("At least one of the conditions is True")
```

# Se – comando pass

Declarações ***if*** não podem estar vazias, mas se por algum motivo você tiver uma declaração ***if*** sem conteúdo, coloque a instrução ***pass*** para evitar um erro

```
a = 33
b = 200

if b > a:
  pass
```

# 04

# Estruturas de condição do tipo “faça caso”

- Match

# Match - case

Seja a seguinte estrutura condicional, onde cada condição é mutuamente exclusiva:

```
if (condição 1):
    Instrução
elif (condição 2):
    Instrução
elif (condição 3):
    Instrução
elif (condição 4):
    Instrução
```

# Match - case

A partir da versão 3.10, a estrutura **Match case** pode ser utilizada quando temos situações mutualmente exclusivas, em que, se uma instrução for executada, as demais não serão:

```
def get_color(color_code):
    match color_code:
        case '#FF0000':
            print("Red")
        case 3093151:
            print("Blue")
        case False:
            print("Not a color")
        case None:
            print("None")


get_color('#FF0000')
```

# Match - case

Seja a seguinte estrutura condicional, onde cada condição é mutuamente exclusiva:

```
if (condição 1):
    Instrução
elif (condição 2):
    Instrução
elif (condição 3):
    Instrução
elif (condição 4):
    Instrução
```

```
match (condição):
    case (critério 1):
        Instrução
    case (critério 2):
        Instrução
    case (critério 3):
        Instrução
    case (critério 4):
        Instrução
```

# Obrigada!